Accelerating the quality control of genetic sequences through stream processing

Óscar Castellanos-Rodríguez
Universidade da Coruña, CITIC
A Coruña, Spain
oscar.castellanos@udc.es

Roberto R. Expósito
Universidade da Coruña, CITIC
A Coruña, Spain
roberto.rey.exposito@udc.es

Juan Touriño
Universidade da Coruña, CITIC
A Coruña, Spain
juan.tourino@udc.es

ABSTRACT

Quality control of DNA sequences is an important data preprocessing step in many genomic analyses. However, all existing parallel tools for this purpose are based on a batch processing model, needing to have the complete genetic dataset before processing can even begin. This limitation clearly hinders quality control performance in those scenarios where the dataset must be downloaded from a remote repository and/or copied to a distributed file system for its parallel processing. In this paper we present SeQual-Stream, a Big Data tool that allows performing quality control on genomic datasets in a fast, distributed and scalable way. To do so, our tool relies on the Apache Spark framework and the Hadoop Distributed File System (HDFS) to fully exploit the stream paradigm and accelerate the preprocessing of large datasets as they are being downloaded and/or copied to HDFS. The experimental results have shown significant improvements when compared to a batch processing tool, providing a maximum speedup of 2.7x.

CCS CONCEPTS

• **Applied computing** → **Bioinformatics**; *Computational genomics*; • **Computing methodologies** → MapReduce algorithms;

KEYWORDS

Big Data, Stream processing, Next Generation Sequencing (NGS), Quality control, Apache Spark

ACM Reference Format:
Óscar Castellanos-Rodríguez, Roberto R. Expósito, and Juan Touriño. 2023. Accelerating the quality control of genetic sequences through stream processing. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-April 2, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555776.3577785

1 INTRODUCTION

Obtaining DNA sequences from living beings is usually the first step in the studies developed by biologists and bioinformaticians. The continuous development of Next Generation Sequencing (NGS) technologies [14] has led to a vertiginous increase in the amount of available genomic data. Hundreds of millions of sequences (so-called reads) can now be generated in a single experiment at a drastically reduced cost. However, the accuracy of current NGS platforms is not high in all cases. The quality of downstream analyses may be affected because of the artifacts introduced in some DNA fragments during the sequencing process [6, 11], regardless of the sequencing platform. Therefore, quality control is an essential preprocessing step for raw NGS data [9], removing or modifying those input reads that are not considered useful.

In this paper we introduce SeQual-Stream, a parallel tool implemented in Java that allows performing multiple quality control operations (e.g., trimming, filtering) on large genomic datasets in a distributed and scalable way. To do so, it takes full advantage of the Apache Spark Big Data framework [21] together with the Hadoop Distributed File System (HDFS) [16]. All existing parallel quality control tools operate on a batch processing model, which means that they require the entire input dataset before any processing can begin. This poses a performance constraint, as downloading the data from a remote repository and copying them to a distributed file system such as HDFS for parallel processing are costly operations that significantly delay the start of the quality control. This problem is especially relevant in the NGS context as the size of the genomic datasets is continuously increasing, which demands more efficient processing modes. To overcome this issue, SeQual-Stream has been implemented upon the Spark Structured Streaming API [19], in order to apply the quality control operations to the input reads as the data are being downloaded from a remote location (e.g., a web repository) and/or copied to HDFS. Up to our knowledge, this is the first quality control tool that can exploit the stream processing model to accelerate the preprocessing of raw NGS datasets.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes briefly the implementation of SeQual-Stream. Section 4 evaluates its performance compared to a quality control parallel tool that processes data in batch mode. Finally, Section 5 concludes the paper.

2 RELATED WORK

There is a wide variety of tools within the context of bioinformatics. Many of them follow a batch model, such as CloudEC [3] for error correction or BigBWA [1] for sequence alignment, both of them relying on the Apache Hadoop Big Data framework [17].

Focusing on quality control tools, all existing approaches are based on batch processing. Examples of such tools are FASTX-Toolkit [8] and PRINSEQ [15], which do not even support parallel processing, whereas QC-Chain [23] and PRINSEQ++ [2] provide such support through multithreading, and so their scalability is limited to a single node. SeQual [7] is a quality control tool capable of scaling out across a cluster of nodes by relying on Spark, greatly

SAC '23, March 27-April 2, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03.
https://doi.org/10.1145/3555776.3577785

enhancing performance compared to previous solutions. Nevertheless, SeQual is still limited by the batch processing operation mode it is based on.

3 IMPLEMENTATION

SeQual-Stream is a parallel Java tool that provides a wide set of operations to apply quality control and preprocessing on raw NGS data. These operations are grouped into three different categories depending on the functionality they provide: (1) single filters, responsible for discarding input sequences that do not meet a certain criteria (e.g., sequence length), evaluating each sequence independently of the others; (2) trimmers, operations that trim certain sequence bases at the beginning or end; and (3) formatters, operations to change the format of the input dataset (e.g., from DNA to RNA). The tool can receive as input single- or paired-end datasets [5], supporting FASTQ [4] and FASTA [22] formats. The input files can be stored in HDFS or locally. The datasets may be complete or in the process of being downloaded from a remote server, since SeQual-Stream can process data as new sequences continue to arrive. Note that in the case of having the complete files stored locally, they are also processed in a streaming way as they are copied to HDFS.

The overall dataflow of the tool can be divided into three main stages as follows:

(1) Reading of the input dataset(s), which may be stored in HDFS or locally and may be in the process of being downloaded.
(2) Processing of the sequences by applying the quality control operations configured by the user.
(3) Writing of the results to the output files using the path specified by the user.

The next sections provide more details about the implementation of each stage.

3.1 Reading of the input datasets

The objective of the first stage is the creation of a Spark Dataset [18] that represents in a relational table the sequences to be processed in the next stage. Basically, Structured Streaming operates by indicating a directory in HDFS to be monitored and processing the files as they are written to it. The problem is that once the available data of a certain file has been processed, such file is not processed again even if it is updated with new content. So, it cannot be used to process large files that are still in the process of downloading. To overcome this issue, the proposed solution consists of creating a previous stage in charge of reading the input dataset and generating new files formed by subsets of the input data (called "subfiles") that Structured Streaming is able to process. Note that this reading stage works iteratively. For example, if there is a subset of sequences downloaded at a given time from a certain dataset, this stage will do a first iteration to store those sequences in a new subfile on HDFS so that Structured Streaming can process it, and then it will wait for the remaining sequences to be downloaded. After a few seconds, it will recheck the state of the input dataset and, if new data is found, the procedure is repeated through a second iteration, generating a new subfile with new sequences to be processed. In order to speed up this process, the reading is divided and performed in parallel in one of the cluster nodes through multithreading.

It is important to remark that only complete sequences are copied to subfiles. If no more data is available at a given time and the last sequence is incomplete, only the complete sequences before the last one (if any) are copied while waiting for new data to arrive. Therefore, the copy operations cannot be done on a line-by-line basis, since it is necessary to evaluate if sequences are complete as they are represented in multiple lines (e.g., at least two for FASTA format). This process is even more complex for paired-end datasets, where there are two input files to be downloaded. In addition to copy only complete sequences, it must be done synchronously in both files because one of them may have more available data than the other as download speeds may differ. The solution to this issue is reading the sequences in pairs (i.e., only if both are complete) and copy them together within the same subfile.

Once new subfiles are copied to HDFS, Structured Streaming is able to automatically detect them to allow SeQual-Stream creating a Spark Dataset of sequences. Although Spark supports several common file formats (e.g., JSON, CSV), sequence formats cannot be read directly, and so the standard text-based file format provided by Spark must be used. In order to differentiate each sequence (or pair of sequences) and separate them correctly as rows in the Spark Dataset, an unambiguous separator is added to each one.

3.2 Processing of the sequences

The next stage of the pipeline is the processing of the sequences contained on a Spark Dataset by applying the quality control operations selected by the user. The first group of operations consists of 12 single filters that were implemented using the Spark's *filter* method. Each operation implements the corresponding boolean function to discard those sequences that do not meet a certain criteria. For example, the Length filter evaluates whether the size of the sequence is smaller and/or larger than an upper and/or lower limit configured by the user.

The second and third group of operations (10 trimmers and 3 formatters, respectively) were implemented using the Spark's *map* method, which processes the Dataset by applying a specific function to each element (i.e., sequence). For example, TrimLeft trims a given number of bases from each sequence (and their quality scores if applicable) starting from the left using the Java *substring* method.

Note that the quality control operations are performed as new sequences are loaded into the Spark Dataset, so that the processing stage efficiently overlaps with the reading of the input dataset.

3.3 Writing of the results

After a certain set of operations is performed over the sequences, they must be written back to HDFS. Due to the distributed nature of Spark, the output sequences are written throughout different output files, the main issue being how to keep these sequences in the same order that the input. When persisting Spark data, the different output files (or "parts") are named in alphabetical order so that the original order is maintained. For example, the first part file ("part-0000") may contain the sequences from 1 to 100, the second one ("part-0001") from 101 to 200, and so on. The problem arises when using Structured Streaming, since Spark processes each subfile independently and does not preserve an alphabetical naming order between parts generated from different subfiles. A

Table 1: Hardware characteristics of the cluster nodes

CPU Model	2 × Intel Xeon E5-2660 Sandy Bridge-EP
CPU Speed/Turbo	2.20 GHz/3.0 GHz
#Cores per node	16
#Threads per node	32
Cache L1/L2/L3	32 KiB/256 KiB/20 MiB
Memory	64 GiB DDR3 1600 Mhz
Disk	1 × HDD 1 TB SATA3 7.2K rpm
Networks	InfiniBand FDR & Gigabit Ethernet

Table 2: Characteristics of the datasets

Dataset	SRR567455	SRR11442499
Tag	SRR56	SRR114
Organism	Homo sapiens	Homo sapiens
#Reads	2 x 251.9 M	2 x 250.3 M
Read length	76 base pairs	99 base pairs
Size	2 x 45 GiB	2 x 62 GiB

naive solution would be using the write timestamp of each part file, since the first sequences should be processed and written before the following ones. However, this rule is not consistent because there is no guarantee that Structured Streaming processes the subfiles in the same order they were created.

Our solution consists of embedding a custom timestamp in each sequence during the reading stage so that the order is set from the very beginning. More specifically, our tool must be able to differentiate each generated subfile during such stage. To do so, a numeric code that represents each subfile is used as timestamp. Therefore, SeQual-Stream actually processes a Spark Dataset containing sequences tagged with a timestamp. Right before writing the results to HDFS, this dataset is separated into two columns: the sequences themselves and their timestamps. This approach allows writing the results partitioned by the timestamp column, an operation that consists of gathering the part files containing sequences with the same timestamp into the same output directory. Those parts are already sorted alphabetically, thus the global order is ensured.

Regarding the writing operation itself, it is done through a Spark object called "StreamingQuery" that remains in a loop as long as there is data to be written. This loop ends when the reading stage sends a specific signal meaning that there is no more input data, and when the StreamingQuery has no pending data to write.

4 PERFORMANCE EVALUATION

The experimental evaluation has been conducted on a 17-node cluster consisting of one master and 16 worker nodes. The hardware characteristics of the nodes are summarized in Table 1. Two publicly available FASTQ datasets have been evaluated, obtained from the Sequence Read Archive (SRA) [10, 13], a public repository of genomic data belonging to the National Center for Biotechnology Information (NCBI) [12, 20]. These datasets present a paired-end layout, so they consist of two FASTQ files (single-end experiments use one file). Their characteristics are summarized in Table 2.

The evaluation has been carried out comparatively with SeQual using two representative quality control operations:

- QUALITY: a single filter that filters sequences based on an indicated maximum and/or minimum mean quality threshold. A minimum quality of 25 was used in the experiments.
- TRIMRIGHTP: a trimmer that trims sequences according to an indicated percentage of the total number of bases starting from the right. A 10% trimming was used in the experiments.

During the experiments the complete input dataset was already stored in the master node (i.e., it was previously downloaded). In this scenario, SeQual first requires to copy the dataset to HDFS for processing it, since data processing can only begin once the complete dataset was copied. However, SeQual-Stream is able to read it directly from the local file system of the master node and start its processing while it is being copied to HDFS.

4.1 Analysis of the results

Table 3 shows the execution times of both tools for all the scenarios under evaluation. Note that the results for SeQual take into account the time required to copy the dataset before starting its processing. As can be observed, the speedups obtained by our tool range from a minimum of 1.00x up to a maximum of 2.70x, being the average speedup around 1.45x. In general, the speedup is usually higher when using a small number of nodes, and it tends to converge to 1 when using 16 worker nodes. The main reason is that there comes a point where there is so much computational power and disks on which to spread the write operations, that the processing and writing of the results are fast enough and the performance limiting factor is the speed of copying the input files, which will be similar for both tools.

Overall, the speedups tend to increase noticeably for paired-end experiments compared to single-end ones. In fact, all speedups greater than 2x are achieved in the paired-end mode. It is important to remark that this mode involves copying and processing twice as much data as single-end. As the amount of input data increases significantly, the time required to copy them to HDFS, process them with Spark and write their parts to HDFS also increases proportionally. Therefore, parallelizing all this process through a stream model is very beneficial and is precisely what was sought after with the development of this tool.

5 CONCLUSIONS

The large amount of genomic data generated by modern NGS technologies reinforces the need for bioinformatics tools capable of reducing the time required for processing them as much as possible. In this paper we have presented a Big Data tool for quality control of raw NGS datasets which seeks to reduce processing times through the use of Apache Spark and its Structured Streaming API. This combination allows our tool to take full advantage of distributed-memory systems such as clusters and to further accelerate quality control by overlapping data processing with downloading and/or HDFS copying operations.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Science and Innovation of Spain (PID2019-104184RB-I00 / AEI / 10.13039 / 501100011033),

Table 3: Runtimes (in seconds) of SeQual and SeQual-Stream using 1, 4 and 16 worker nodes when performing quality control operations on different single- and paired-end datasets. The last column shows the speedup of SeQual-Stream over SeQual

Operation	Dataset	Mode	Nodes	SeQual	SeQual-Stream	Speedup
QUALITY	SRR56	Single	1	1386	927	1.49
			4	780	595	1.31
			16	522	513	1.02
		Paired	1	5312	1967	2.70
			4	2043	1102	1.85
			16	1024	940	1.09
	SRR114	Single	1	2239	1918	1.17
			4	1242	941	1.32
			16	723	689	1.05
		Paired	1	6983	4082	1.71
			4	3126	1688	1.85
			16	1452	1306	1.11
TRIMRIGHTP	SRR56	Single	1	1682	1062	1.58
			4	765	680	1.13
			16	521	523	1.00
		Paired	1	5850	2259	2.59
			4	2280	1463	1.56
			16	1051	961	1.09
	SRR114	Single	1	2855	1723	1.66
			4	1050	909	1.16
			16	729	688	1.06
		Paired	1	5960	4119	1.45
			4	3329	1969	1.69
			16	1420	1325	1.07

and by Xunta de Galicia and FEDER funds of the European Union (Centro de Investigación de Galicia accreditation 2019-2022, ref. ED431G 2019/01; Consolidation Program of Competitive Reference Groups, ref. ED431C 2021/30).

REFERENCES

[1] José M. Abuín, Juan C. Pichel, Tomás F. Pena, and Jorge Amigo. 2015. BigBWA: Approaching the Burrows–Wheeler aligner to Big Data technologies. *Bioinformatics* 31, 24 (2015), 4003–4005.

[2] Vito Adrian Cantu, Jeffrey Sadural, and Robert Edwards. 2019. PRINSEQ++, a multi-threaded tool for fast and efficient quality control and preprocessing of sequencing datasets. *PeerJ Preprints* 7, Article e27553v1 (2019), 3 pages.

[3] Wei-Chun Chung, Jan-Ming Ho, Chung-Yen Lin, and Der-Tsai Lee. 2017. CloudEC: A MapReduce-based algorithm for correcting errors in next-generation sequencing big data. In *Proceedings of the 2017 IEEE International Conference on Big Data (IEEE BigData 2017)*. Boston, MA, USA, 2836–2842.

[4] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. 2009. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research* 38, 6 (2009), 1767–1771.

[5] Columbia Genome Center. [n. d.]. Genome sequencing: Defining your experiment. Retrieved Dec. 2022 from https://systemsbiology.columbia.edu/genome-sequencing-defining-your-experiment

[6] Robert C. Edgar and Henrik Flyvbjerg. 2015. Error filtering, pair assembly and error correction for next-generation sequencing reads. *Bioinformatics* 31, 21 (2015), 3476–3482.

[7] Roberto R. Expósito, Roi Galego-Torreiro, and Jorge González-Domínguez. 2020. SeQual: Big Data tool to perform quality control and data preprocessing of large NGS datasets. *IEEE Access* 8 (2020), 146075–146084.

[8] Assaf Gordon and Gregory J. Hannon. 2010. FASTX-Toolkit: FASTQ/A short-reads pre-processing tools. Retrieved Dec. 2022 from http://hannonlab.cshl.edu/fastx_toolkit

[9] Binsheng He et al. 2020. Assessing the impact of data preprocessing on analyzing Next Generation Sequencing data. *Frontiers in Bioengineering and Biotechnology* 8, 817 (2020), 1–12.

[10] Yuichi Kodama, Martin Shumway, and Rasko Leinonen. 2011. The sequence read archive: Explosive growth of sequencing data. *Nucleic Acids Research* 40, D1 (2011), D54–D56.

[11] André Minoche, Juliane Dohm, and Heinz Himmelbauer. 2011. Evaluation of genomic high-throughput sequencing data generated on Illumina HiSeq and Genome Analyzer systems. *Genome Biology* 12, R112 (2011), 1–15.

[12] National Center for Biotechnology Information. [n. d.]. NCBI. Retrieved Dec. 2022 from https://www.ncbi.nlm.nih.gov/

[13] National Center for Biotechnology Information. [n. d.]. The Sequence Read Archive (SRA). Retrieved Dec. 2022 from https://www.ncbi.nlm.nih.gov/sra

[14] Kathryn A Phillips. 2018. Assessing the value of next-generation sequencing technologies: An introduction. *Value in Health* 21, 9 (2018), 1031–1032.

[15] Robert Schmieder and Robert Edwards. 2011. Quality control and preprocessing of metagenomic datasets. *Bioinformatics* 27, 6 (2011), 863–864.

[16] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. 2010. The Hadoop distributed file system. In *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST 2010)*. Incline Village, NV, USA, 1–10.

[17] The Apache Software Foundation. [n. d.]. Apache Hadoop. Retrieved Dec. 2022 from https://hadoop.apache.org

[18] The Apache Software Foundation. [n. d.]. Spark SQL, DataFrames and Datasets guide. Retrieved Dec. 2022 from https://spark.apache.org/docs/latest/sql-programming-guide.html

[19] The Apache Software Foundation. [n. d.]. Structured Streaming programming guide. Retrieved Dec. 2022 from https://spark.apache.org/docs/3.1.1/structured-streaming-programming-guide.html

[20] David L. Wheeler et al. 2007. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research* 36, suppl_1 (2007), D13–D21.

[21] Matei Zaharia et al. 2016. Apache Spark: A unified engine for Big Data processing. *Commun. ACM* 59, 11 (2016), 56–65.

[22] Zhang Lab. [n. d.]. What is FASTA format? Retrieved Dec. 2022 from https://zhanggroup.org/FASTA/

[23] Qian Zhou, Xiaoquan Su, Anhui Wang, Jian Xu, and Kang Ning. 2013. QC-Chain: Fast and holistic quality control method for next-generation sequencing data. *PLOS ONE* 8, 4, Article e60234 (2013), 10 pages.